Data products review

or, “what have you done with my data”

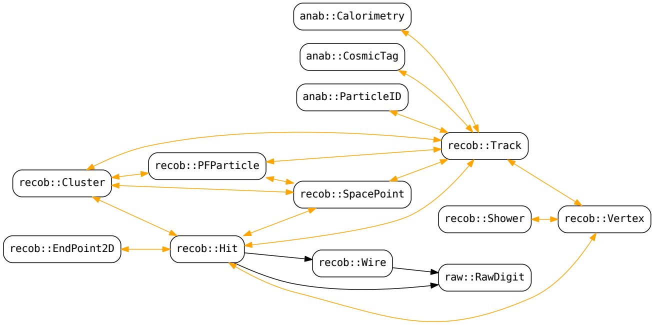
Gianluca Petrillo, Erica Snider

MicroBooNE Collaboration Meeting, December 17th , 2014

Data products review

`LArSoft` comprises about 40 classes that can be serialized into ROOT files (“data products”).

Here the largest block of interconnected products is shown:



Class relationships are expressed through *data members* (as `art` pointers) and *associations* (a separate data product).

The review process

These products are currently being reconsidered:

- do they still reflect useful concepts?
- do they still meet experiments' current needs?
- do they look flexible enough to accommodate future needs?
- can they read outside an `art` environment?

The review process is such organized:

1. there is a lot of discussions with code authors
2. we collect the ideas and synthesize proposals
3. proposals are evaluated by a specific meeting *with experiments and framework representatives*
4. often, it's `GOTO 1`
5. finally, it's time to implement them

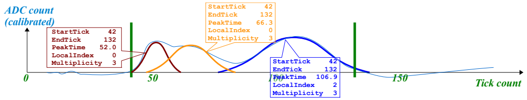
Typical changes

These are the typical actions proposed:

- fix major design problems
  e.g. dark tales are whispered about Bézier tracks
- remove framework dependencies, make the data readable in any environment
  e.g. goodbye `Hit::Wire()`, and constructors using `art::Ptr`
- incorporate data of frequent use and slow to retrieve
  e.g. which channel ID for this `Wire`?
- remove data members that are unused or ill-defined
  e.g. `Hit::SigmaStartTime()`
- add fashionable new data members
  e.g. `Track::Quality()`, `Hit::SummedADC()`
- rename methods to some more sensible name
  e.g. `Hit::StartTime()` ⇒ `Hit::StartTick()`

First example: `recob::Hit` (I)

`recob::Hit`: charge deposition from a particle on a wire:



- `StartTime()` and `EndTime()`, start and end of the hit on the wire, are actually in units of TDC counts; so we
  - define them as enclosing the portion of wire used to fit the hit
  - call them `StartTick()` and `EndTick()`
  - make them integrals
  - remove their uncertainties
- there may be multiple hits in the same time range: we allow each hit to know
  - how many hits are reconstructed in that range (`Multiplicity()`)
  - which of them this one is (`LocalIndex()`)
- `fHitSignal` is a public member that is never filled; we define it as the content of the wire in the time range... and fill it, too

First example: `recob::Hit` (II)

For example, here are some changes on `recob::Hit`:

- add a channel ID (default value `raw::InvalidChannelID`)
- `Charge(false)` used to return the hit signal integral, `Charge(true)` its peak amplitude; so now `Charge` is gone, enter `Integral()` and `PeakAmplitude()`; with uncertainties
- add a sum of the ADC values belonging to the hit (`SummedADC()`)
- add hit width (`RMS()`)
- remove `art` pointers to raw digits and wire from data members and as constructor arguments, replace them with associations to raw digits and wire

Additional functions, external to the data product, may be provided to replace the functionality of the removed constructors.

The other example: `recob::Track`

The track object (`recob::Track`) has some problems:

- lacks a definition about density of trajectory points
- contains elements that are algorithm-dependent
- is able to host Bézier tracks only by a (ugly) hack

A proposal, under discussion, consists in:

- leaving in `recob::Track` only general track properties: start and end point, intermediate trajectory points, track quality
- prescribing by policy the trajectory point density
- delegating the continuous representation to another object (`recob::Trajectory`)
- moving the momentum estimation into a new object (`recob::Momentum`)
- binding all these objects with associations

Wait a moment... won't this break the existing data?... yep.

Time line:

- this revolution will render the existing data unreadable, a textbook example of breaking change
- the "breaking" changes take priority; "patchable" changes[1] may be implemented later
- we want MicroBooNE to be able to use the new classes in their next Monte Carlo Challenge (*end of January 2015*)
- aiming to have a number of them ready for test *before* the MicroBooNE Reconstruction Workshop (*January 5th , 2015*)

This is a *very ambitious* goal.

[1] Changes that can be made in a non-breaking way, given enough time and pain.

Status

- I have started working on the changes already approved
  raw digits, wires, hits
- the code is in a branch `feature/DataProductRevision`
- a itemized status is available on the web: `https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/DataProductsArchitectureActionItems`
- code will change as we change our mind

My working model is being:

- deliver something compiling and well-documented
  - ⇒ changes in the 10 `LArSoft` repositories and the 4+ experiment repositories
- rely on the algorithm authors to properly fill the new data

Outlook

- we are reviewing the main reconstruction data products
- we have started with the implementation of the changes already agreed upon
- we ask the help of algorithm authors to complete the work
- we also need the experiments to actively scrutiny the changes under discussion
- a shared strategy and time line needs to be discussed and blesses by the experiments

Additional material

Example of direct use of data in another framework

Say that your favourite framework has a data model where all data classes inherit from `TDataObject` (may be even ROOT's `TObject`).
You can define a base class:

```
#include "ArtlessFW/TDataObject.h"
#include "RecoBase/Hit.h"

template <typename Data>
class TDataWrapper: public Data, public TDataObject { /* ... */ };

namespace wrapped {
  class Hit: public TDataWrapper<recob::Hit> { /* ... */ };
}
```

- `wrapped::Hit` includes the very interface of `recob::Hit`
- ... plus additional methods that you care to define specifically for it
- you are using the original `lardata/RecoBase/Hit.h[.cxx]`
- you might even read directly from the art tree into `wrapped::Hit`

* I haven't tried the limits of this model, but it might even work
(if I were sure, this slide would not be in the appendix!)

The current proposal: raw digits and wires

The review has so far focused on six data products.

`raw::RawData`: straight readout from DAQ

- removal of pedestal information, *still under discussion*

`recob::Wire`: calibrated and filtered signal on a channel

- removal of `art` pointers
- addition of channel ID member
- removal of `SignalType()`

The current proposal: hits and clusters

`recob::Hit`: deposition from a particle on a wire

- removal of `art` pointers
- removal of unused information
- addition of hit width (RMS)
- more precise definition of start and end times (now defined in TDC tick units), peaks and integrals

`recob::Cluster`: set of hits from the same particle

- removal of unused information (dQ/dx)
- addition of shower/track discriminating variables
- more things being discussed

The current proposal: tracks and 2D end-points

`recob::Track`: reconstructed path of a ionizing particle

A lot of discussion ongoing... separate in multiple classes:

- a discrete track, result of the tracking fit
- a trajectory, continuous representation of the particle path
- a momentum describing the initial impulse of the particle

`recob::EndPoint2D`: a point on a wire plane

Still under discussion. Split into:

- a geometric object: just two coordinates
- an object result of a reconstruction algorithm (with goodness of fit etc.)